PROGRAMAÇÃO DINÂMICA

- *Cria-se uma matriz de distâncias com*
  - uma coluna por cada símbolo da sequência de saída
  - uma linha por cada símbolo da sequência de entrada
  - cada célula `distância[j,i]` contém a distância entre os primeiros `i` caracteres da saída e os primeiros `j` caracteres da entrada

	9	n										
	8	o										
	7	i										
	6	t										
i	5	n										
	4	e										
	3	t										
	2	n										
	1	i										
	0	#										
			#	e	x	e	c	u	t	i	o	n
			0	1	2	3	4	5	6	7	8	9
			j									

PRINCÍPIOS

- *Cada célula pode ser calculada usando uma função das células que a rodeiam, sendo cada célula preenchida uma única vez*

- *O valor de cada célula é calculado como o mínimo de três passos alternativos:*
  - três possibilidades para chegar ao estado [j,i]:
    - por inserção de um carácter
    - por substituição de um carácter
    - por remoção de um carácter

$$\min \begin{Bmatrix} \text{distância}[j-1,i] + \text{custo_Inserção}(\text{Saída}_i) \\ \text{distância}[j-1,i-1] + \text{custo_Substituição}(\text{Saída}_i, \text{Entrada}_j) \\ \text{distância}[j,i-1] + \text{custo_Remoção}(\text{Entrada}_j) \end{Bmatrix}$$

ALGORITMO

```
função número_mínimo_edições(saída, entrada) devolve número_mínimo

n ← comprimento(saída)
m ← comprimento(entrada)

cria a matriz distância[n+1, m+1]

distância[0,0] ← 0

para cada coluna j de 1 a n faz
  distância[j,0] ← distância[j-1,0] + custo_inserção(saída_j)

para cada linha i de 1 a n faz
  distância[0,i] ← distância[0,i-1] + custo_remoção(entrada_i)

para cada coluna j de 1 a n faz
  para cada linha i de 1 a m faz
    distância[j,i] ← min(distância[j-1,i]   + custo_inserção(saída_j),
                         distância[j-1,i-1] + custo_subs(saída_j,entrada_i)
                         distância[j,i-1]   + custo_remoção(entrada_i) )
```


EXEMPLO